

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
14 December 2000 (14.12.2000)

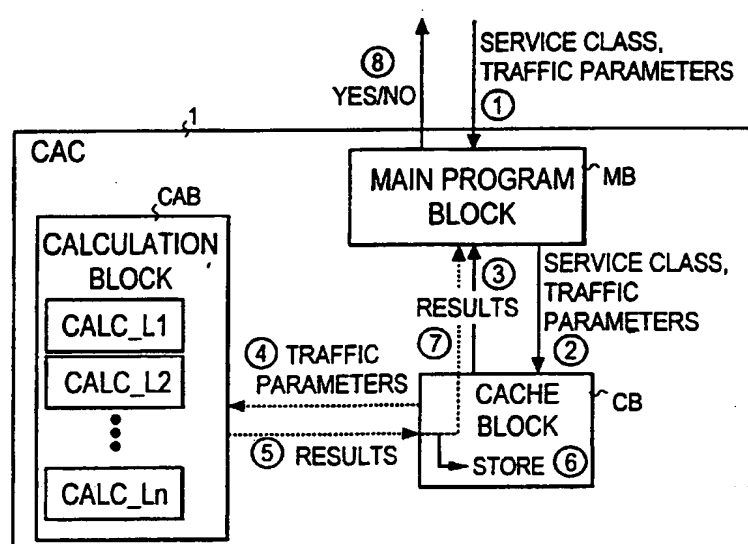
PCT

(10) International Publication Number  
**WO 00/76254 A1**

- (51) International Patent Classification<sup>7</sup>: H04Q 11/04, H04L 12/56
- (21) International Application Number: PCT/FI00/00345
- (22) International Filing Date: 20 April 2000 (20.04.2000)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:  
991309 8 June 1999 (08.06.1999) FI  
992450 15 November 1999 (15.11.1999) FI
- (71) Applicant (for all designated States except US): NOKIA NETWORKS OY [FI/FI]; Keilalahdentie 4, FIN-02150 Espoo (FI).
- (72) Inventor; and
- (75) Inventor/Applicant (for US only): KINNUNEN, Matti [FI/FI]; Haavikkotie 15-17 A 12, FIN-00120 Helsinki (FI).
- (81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).
- Published:  
— With international search report.  
— Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.

[Continued on next page]

(54) Title: CONNECTION ADMISSION IN A COMMUNICATIONS NETWORK



(57) Abstract: The invention relates to Connection Admission Control (CAC). In response to a connection request received from a traffic source, a set of input data is specified and when necessary a set of results is calculated by means of the input data. A total capacity requirement is then estimated on the basis of said set of results and the capacity requirement of existing connections. In order to obtain a connection admission control which can be flexibly adapted to the various constantly changing network environments, the sets of input data and the corresponding sets of results relating to previous requests are stored in a cache. When a new connection request arrives, a similar set of data is searched for in the cache. When it is found, the stored set of results is used directly for estimating the total capacity, and the calculation is omitted.

WO 00/76254 A1

WO 00/76254 A1

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684

## Connection admission in a communications network

### Field of the invention

This invention relates to Connection Admission Control (CAC) in  
5 communications networks, especially in ATM-based broadband networks.

### Background of the invention

ATM (Asynchronous Transfer Mode) is a known connection-oriented  
packet-switching technique, which has been selected by the international  
10 organization for telecommunications standardization ITU-T as the target  
transfer mode solution for implementing a broadband multimedia network (B-  
ISDN). In these networks many traffic sources are bursty, whereby the  
transmission capacity required at some moment is high and the capacity  
required at another moment is low. Bursts may be described as a  
15 phenomenon whereby a set of cells (a burst) arrives at short intervals and the  
following set (burst) arrives after a relatively long time. Since different  
connections need different capacities and the need varies quickly, statistical  
multiplexing is used in an ATM-based network. The statistical characteristics of  
bursty traffic sources are utilized in statistical multiplexing: when a large set of  
20 traffic sources are combined, the combined traffic behaves in a more stable  
manner than the individual sources, and although the transmission speed of  
an individual traffic source varies, the combined transmission speed of  
mutually independent individual traffic sources is almost constant. By using  
statistical multiplexing, it is possible to serve more connections with the same  
25 resources, i.e. the utilization rate of the network can be increased.

In spite of statistical multiplexing, congestion and overload will still  
occur in the network, caused both by unpredictable statistical variations in the  
traffic and by failure situations occurring in the network. Since it is impossible  
to know with sufficient accuracy the services to be provided, the volume of  
30 traffic brought about by the services, and the exact characteristics of the traffic  
sources, the occurrence of congestion is unavoidable in a network. The  
purpose of traffic and congestion control is to protect the network and the user  
so that the desired quality of a service is achieved.

Traffic control is intended to prevent the occurrence of congestion  
35 situations, while congestion control reacts to congestion situations detected in  
the network. From the point of view of the quality of service (QoS), most

significant are the traffic control functions which try to prevent congestion situations in advance, of which Connection Admission Control (CAC) is the most important preventive traffic control method. CAC is defined as the set of procedures taken by the network during the call (virtual connection) set-up phase, or during the call re-negotiation phase, to determine whether a connection request can be accepted or rejected. CAC accepts a connection only when sufficient resources are available at each successive link in the circuit, and when the new connection will not adversely affect the quality of service of existing connections.

Several different methods of implementing CAC are known. These methods are based either on traffic parameters stated by the user or on traffic measurements taking place in the network element. It is possible to estimate the quality of service or the capacity requirement which can be expected, according to traffic parameters, either with the aid of pre-computed tables as in methods termed "indirect" or by computing in real time based on traffic parameters supplied in the connection request, as in methods termed "direct".

The traffic parameters defined by ATM Forum and describing the inherent characteristics of a traffic source requesting a connection are: Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), Maximum Burst Size (MBS), and Minimum Cell Rate (MCR).

Various requirements, at least some of which are contradictory to one another, must be set for the CAC algorithms determining the acceptability of a connection request. A CAC algorithm should, for example, be as generic as possible in the sense of its applicability to the management of traffic sources of a wide number of different types. The algorithm must also be simple enough, for example, to be applied in real-time so that it will not form a bottle-neck restricting the call processing capacity of the switch. On the other hand, the algorithm should be very efficient in order to utilize the network resources optimally and to guarantee fair treatment of connections of different types.

Advanced and efficient CAC algorithms tend to be computationally expensive. It would be desirable if the CAC algorithm could calculate in advance the extent of resources the connections require. CAC would then be able to decide faster whether a new connection can be accepted or not. However, it is impossible to know in advance the traffic situation at the arrival time of a new connection request. In other words, it is impossible to know in ad-

vance the existing connections and their types, traffic parameters and the quality of service requirements. Thus, CAC cannot calculate in advance, or off-line, the resources needed by the connections existing at the time of a new connection request.

5           As indicated above, when indirect methods are used, the capacity requirement is searched for from a pre-calculated table on the basis of the traffic parameters included in the connection request. The result obtained from the table is then added to the capacity requirement of existing connections to determine whether the connection request can be accepted. Thus, the idea  
10 behind these methods is to minimize the real-time computing needed in the decision-making.

          The drawback of these methods is that the traffic parameters used in the network and the values possible for each parameter must be known in advance when the tables are formed. Therefore, these methods cannot be  
15 flexibly adapted to the various constantly changing network environments. This drawback will be aggravated in future networks where new applications and new terminal equipment are likely to generate traffic with unforeseen characteristics.

## 20           **Summary of the invention**

          The purpose of the invention is to eliminate the above-mentioned drawback and to bring about a method allowing fast connection admission without the need to update tables in the network elements when new applications or new terminals are introduced into the network.

25           This goal can be attained by using the solution defined in the independent patent claims.

          The basic idea of the invention is to combine the above-mentioned direct method, i.e. real-time computing, with caching by storing a certain number of previous connection requests and their results and using the cached  
30 information when a new connection request similar to any of the said previous requests arrives. Thus, the idea is to cache the results of those calculations that are likely to be made repeatedly and that can be calculated without knowledge of the currently existing connections. In this way CAC decision-making can be accelerated significantly. Every time CAC receives a new connection  
35 request, it checks from its cache whether a similar connection request has

occurred in the recent past. If so, CAC uses the cached results and if not, CAC uses the appropriate algorithm to calculate the results.

By means of the solution according to the invention, CAC decision-making can be speeded up without the need to know the traffic parameters in advance or to take extra measures when the network environment changes.

Within a single network element including several sub-units, it is often preferable to implement CAC so that the decision whether a connection request can be accepted is made separately for each sub-unit. Thus, each sub-unit can be provided with a CAC process of its own to decide on the acceptability for the part of the sub-unit concerned, in which case the connection request must pass through several CAC processes (i.e. through several sub-units) before the entire network element can accept it. In a preferred embodiment of the invention which is intended for this kind of implementation, one CAC processing unit sends to the next CAC processing unit, together with the input data, a flag indicating whether the process found cached results and an identifier identifying the results stored in the cache. In this way only the first of the several CAC units has to search for the results in the cache. This makes a CAC for the entire network element faster and computationally less demanding.

#### **Brief description of the drawings**

In the following, the invention and its preferred embodiments are described in closer detail with reference to examples shown in the appended drawings, wherein

Figure 1 is a block diagram of a network environment in which the method can be used,

Figure 2 illustrates the functional blocks carrying out the method in accordance with the invention in a switch,

Figure 3 illustrates the basic embodiment of the data structure used for caching,

Figure 4 illustrates another embodiment of the data structure shown in Figure 3,

Figure 5 illustrates a switch in which connection admission control is performed by several successive CAC processes, and

Figure 6 illustrates the operation of successive CAC processes.

### Detailed description of the invention

ATM Forum has specified five different service classes which relate traffic characteristics and the quality of service requirements to network behavior. These service classes are: constant bit rate (CBR), real-time variable bit rate (rt-VBR), non-real time variable bit rate (nrt-VBR), available bit rate (ABR), and unspecified bit rate (UBR). Each class represents connections which have homogeneous traffic pattern characteristics and QoS requirements. The above-mentioned traffic parameters are used, together with what are termed QoS parameters, to define the service classes. The QoS parameters are: Cell Delay Variation (CDV), Maximum Cell Transfer Delay (Max CTD), and Cell Loss Ratio (CLR).

Figure 1 illustrates an example of the network environment in which the present invention can be implemented. The network comprises a plurality of interconnected (ATM) switches having CAC units (1) which implement the CAC procedure described above. Thus, the CAC units ensure that a connection request received from one of the traffic sources (TE1 to TE5) is accepted only when sufficient resources are available to set-up the connection at its required QoS without degrading the QoS of existing connections. When a connection request arrives in an access node of the network, the node determines the parameters describing the connection or the source and supplies them as an input to the CAC unit. A connection request received from a traffic source includes typically at least the service class required and the traffic parameters describing the source, as shown in connection with terminal TE3 in Figure 1. Thus, at least part of the input information needed by the CAC unit is normally received with the connection request. Part of the said information can be pre-stored in the network element or calculated there, and part can be received from elsewhere than the traffic source, for example from the network management.

Figure 2 is a block diagram of the CAC unit in accordance with the present invention, illustrating the operation of the unit. Various steps of the operation are marked with circled numbers 1 to 8. The CAC unit includes a main program block MB, which receives the input information (such as the service class required and the traffic parameters describing the source) and decides whether to accept or reject the request. Thus, the main program block returns a "yes" or "no" answer in response to the input information supplied to it.

The CAC unit further includes a cache block CB for caching previous requests and their calculated results, as well as a calculation block CAB for performing real time calculation if cached results are not available. When a connection request arrives (step 1), the main program block receives its input information and forwards it to the cache block (step 2). On the basis of this input information, the cache block seeks the results (such as the virtual bandwidth) from its memory. If the cache block finds the results, it supplies (step 3) the information to the main program block, which makes the decision on the basis of the results and the existing connections. For example, if the result is the virtual bandwidth required by the requested connection, the main program block adds this to the virtual bandwidths of existing connections to form an estimate of the total bandwidth needed if the new connection is admitted. The main program block then admits the connection request if the total bandwidth is less than the capacity of the transmission connection (or virtual path), otherwise the request is rejected (step 8).

In this way the request may be processed by the main program block and the cache block only. However, if the cache block does not find results from its memory, it forwards (step 4) the traffic parameters to the calculation block, which calculates the results using the parameters as the input information and supplies these results to the cache block (step 5). The cache block then stores the results together with the corresponding input information into its memory (step 6) so that if a similar connection request arrives in the near future, it can find the result from its memory. Having stored the information, the cache block forwards (step 7) the results to the main program block, which then makes the decision in the above-described manner and accepts or rejects the request (step 8).

As discussed above, when a new connection request arrives from the network, the main program block receives parameters describing the connection or the source, such as the service class and the values of the traffic parameters belonging to that class. The cache block CB includes a data structure illustrated in Figure 3. In this data structure there is first a service class table or array SC, including one element for each service class used in the network. Each element contains a pointer pointing to a class-specific cache array. In this example the number of classes  $L_i$  ( $i=1\dots n$ ) is  $n$ , whereby there are also  $n$  cache arrays  $CA_i$ . Each class-specific cache array contains a certain number of elements, each element containing a pointer pointing to a



linked list LL. Typically, the length of each cache array is between 10 and 20, i.e.  $10 \leq C_i \leq 20$ . The length of two or more cache arrays can be equal, for example, all the cache arrays can include 20 elements.

Each element of an individual cache array points to its own linked list LL, i.e. there is a dedicated linked list for each cache array element. Each linked list contains a certain number of data units DU, each of which, with the exception of the last one, corresponds to one of the traffic parameters in that class. In the example of Figure 3, the first data units in each linked list correspond to Peak Cell Rate (PCR), the second data units to Sustainable Cell Rate (SCR), etc. The number of data units in an individual list depends on the traffic class in question. In the example of Figure 3, the linked lists of the first service class include  $m_1$  parameters, the linked lists of the second service class  $m_2$  parameters, etc, and the linked lists of the  $n^{\text{th}}$  service class  $m_n$  parameters. Each data unit, with the exception of the last one, contains two elements, so that the first element contains a value for the particular traffic parameter and the second element contains a pointer to the next element in the list.

The last data unit in each linked list includes the results for the particular combination of parameter values in that linked list. In other words, the last data unit includes the results obtained from the calculation block when the traffic parameters in that list are used as input data for the calculation. In the example of Figure 3, the results include a certain number of values  $R_i$  in each class. The results can include a value for one or more quantities, such as bandwidth, delay, buffer space requirement, etc.

At the start-up phase of the network element the data structure of Figure 3 can be empty. When the first connection request arrives, the main block supplies the service class information and the traffic parameters to the cache block. As the cache block cannot find the results from its data structure (memory), it calls a class-specific subroutine CALC\_Li in the calculation block and inputs the traffic parameters of the request to the subroutine. The subroutine returns the results to the cache block, which then stores the parameter values and the results to one of the empty lists belonging to that service class. The cache block then sends the results to the main program block, which uses the results to calculate the decision. In this way the data structure is filled up after a certain number of requests have arrived for each service class.

Alternatively, the data structure can be filled up with the most common values of the traffic parameters and their pre-calculated results before the commissioning of the network element.

Thus, in the normal operation state of the CAC unit the data structure of Figure 3 is full of parameter values and results. When a connection request arrives, the search in the cache block starts by examining the service class of the connection request. On the basis of this, the element of array SC corresponding to that particular service class is read. The pointer in this element points to one of the cache arrays, which is then searched through for a similar set of parameters. For the first time the search of the class-specific cache array can begin from the first element of the array, for example. The next time the search can begin from the second element, etc. Nevertheless, the search proceeds from the element of the cache array to the linked list addressed from that element, and further along the list as far as the values of the traffic parameters in the list match the value of the corresponding traffic parameter in the connection request. If the parameter value in the list differs from the value of the corresponding traffic parameter in the request, the search jumps to the next element in the cache array and starts to go through the list associated with that element. Finally, either the results are found at the end of the list with sufficiently similar set of parameters or all the lists have been searched through without a hit. In the latter case, the calculation block calculates the results for the main program block.

The search in the data structure can proceed horizontally as described above, i.e. a list is examined until either a parameter in the list and the corresponding parameter in the request differ or the end of the list is arrived at. Another alternative is to proceed vertically so that the first data units of the lists in that particular class are examined first, then the second data units in those lists in which the value of the first parameter matched that of the parameter in the request, etc.

When the data structure is full, it is updated on a FIFO-basis, i.e. if the current results and parameters are stored in the list addressed from cache array element  $i$ , the next time the results and parameters are stored in the list addressed from element  $i+1$ , etc. In this way the whole content of each class is updated within a certain period. Another alternative is to monitor how frequently each list-specific set of results is used and to replace those sets which are used most infrequently.

Figure 4 illustrates another embodiment of the data structure in the cache block. This embodiment is based on the observation that a certain traffic parameter can have a limited number (N) of possible values per class, whereby that parameter can be taken out from the list and a separate parameter array can be formed for that parameter. Cell Loss Rate (CLR), for example, is a parameter that has at most 20 different values in a single service class. Therefore, the structure can have a separate array CLR\_Ai (i=1..n) for each of the traffic classes (two of them shown in the Figure). The number Ni of elements in an individual array corresponds to the number of different values of Cell Loss Rate in that class.

In this case, the Cell Loss Rate included in the connection request is read without any comparison and the search jumps to the element corresponding to the service class in question and the Cell Loss Rate in the request. For example, if the service class is L2 and the Cell Rate is  $10^{-6}$ , the search jumps directly to the sixth element (which in this example is assumed to correspond to the value  $10^{-6}$ ) in the array of class L2 (as shown in the Figure). Each element in the array includes a pointer which forms the beginning of the linked lists. The lists are similar to those in the first embodiment, except that each list includes one data unit less because the data units corresponding to Cell Loss Rate have been removed from the lists.

In the embodiment of Figure 4, one comparison is saved per each list examined as compared to the embodiment of Figure 3. However, the drawback of the embodiment of Figure 4 is that the number of lists is N-fold, as separate lists must exist for each of the Cell Loss Rate values. Consequently, the memory space requirement is about N-fold as compared to the embodiment of Figure 3 (the number of different Cell Loss Rate values may vary in different classes, whereby the memory space requirement is not exactly N-fold).

In the above description it was assumed that one CAC unit or CAC process attends to connection admission within the switch. However, it is also possible to implement connection admission control as a "multistage" unit or process according to the architecture of the switch, so that there is a CAC unit/process for each "stage" or sub-unit of the switch. Figure 5 illustrates this by showing a generic switch architecture. Subscribers are connected to the switching network SN or to a multiplexer MUX through a subscriber line unit SLU. Multiplexers are used for locally concentrating subscriber traffic within the

switch. The switch is connected to other switches through trunk modules TM. Within the switch, the overall connection admission control can be implemented, instead of one CAC process, by means of several successive CAC processing units described above so that each unit attends to the connection admission control in a certain stage or sub-unit of the switch. Consequently, in a switch according to Figure 5 the overall connection admission control could be implemented, for example, so that the first CAC processing unit (1A) handles connection admission control in the subscriber line units SLU, the second CAC unit (1B) handles connection admission in the multiplexer(s), and the third CAC unit (1C) handles connection admission control in the trunk modules. Thus, depending on the implementation, a connection request may have to pass through several CAC processes within one switch. The request is accepted if each CAC process has accepted it, otherwise it is rejected.

If the connection admission control described above is used in a straightforward manner in a "multistage" CAC implementation, each CAC process has to make a cache lookup in order to check whether it has served similar requests before. This computationally rather demanding process is avoided in a preferred embodiment of the invention which is discussed in the following.

For the sake of simplicity let us assume first that the network element comprises two CAC units, so that a connection request must pass through two successive CAC processes, A and B, within the network element before it can be accepted. Figure 6 illustrates the operation of these two units/processes. Various steps of the operation are marked with circled numbers so that numbers 1 to 6 illustrate the steps associated with a connection request that arrives for the first time and numbers 10 to 15 illustrate the steps associated with a later connection request for which the results can already be found from the cache. As can be seen from the figure, each process has a cache of its own, which is similar to the caches described above. (In the figure, the linked lists have been shortened to fit two successive caches on the paper. For the sake of simplicity, it is further assumed in the figure that each process includes the main program block MB and the calculation block CAB described above.)

When a connection request arrives at process A for the first time (step 1), process A calculates the results and supplies the results to its cache block, as described above (step 2). In this embodiment, however, process A

stores, together with the results, an index which indicates the linked list in question (i.e. the linked list where the results are stored). In the example of Figure 6, the first linked list corresponds to index value one, the second linked list to index value two, etc., and the last linked list corresponds to index value  $l_i$  which is equal to the total number of elements in all the cache arrays. Thus, process A stores a record  $(i, A_1, \dots, A_{k1})$  in the last data unit of one of the linked lists, where  $i$  is the index indicating the linked list where the record is stored and  $A_1, \dots, A_{k1}$  are the results obtained from the calculation block. In this case  $i$  equals 1, since this is the first connection request received by A. If process A then accepts the connection request, it forwards the request to process B (i.e. to CAC unit 1B) by sending a message RM1 to process B (step 3). This message includes field F1, which contains the index that process A used when processing that request, and field F2, which contains a flag indicating whether process A found the results from its cache. In this case, field F2 contains a value F (false), since process A had to calculate the results. The message also includes other data, such as the service class and the traffic parameters (i.e. the input information required by a CAC unit, which was discussed in connection with Figure 2).

When process B (CAC unit 1B) receives message RM1, it updates its cache in the same way as process A did (step 4). The calculated results  $(B_1, \dots, B_{k1})$  are stored, together with the index received, in the linked list corresponding to the index (i.e. in this case in the first linked list). Process B further stores a pointer to the stored record of results  $(1, B_1, \dots, B_{k1})$  in table T1. This table has been formed in advance, before the commissioning of the network element, to be in association with process B, and it is empty before the first connection request arrives. The table contains one cell (CL) for each linked list, and it is indexed by the indices that process B receives from process A. When the table is full, each cell contains a pointer to the record of results in the cache of process B. Thus, since the index received from process A was in this case 1, the pointer PO1 to the record of results  $(1, B_1, \dots, B_{k1})$  is stored in the cell corresponding to the index value 1 (step 5). After this process B either accepts or rejects the connection request (step 6). As the pointer points to the record of results, the value of the pointer word is the same as the value of the pointer word in the second last data unit of the linked list where the record of results was stored.

When a similar request arrives again (step 10), process A finds the results from its cache (step 11). It then sends a message (RM2) to process B (step 12), the message including in field F1 the value of the index found from the cache and a value T (true) in field F2, the latter now indicating to process B that process A found the results from its cache. When process B receives this value in field F2, it knows that it will obtain the results directly by means of the pointer which corresponds to the index received in field F1 of the message. It therefore first reads the pointer corresponding to index value 1 (step 13) and then the results from the memory location addressed by that pointer (step 14).  
10 After this, process B either accepts or rejects the connection request (step 15).

When the caches are full and process A overwrites an entry in its cache, it uses the index corresponding to the linked list in question and supplies this index, together with value F in field F2, to process B. Process B then knows that it must calculate the results, store the record of results at the end of the linked list corresponding to the received index value, and store a pointer into that cell of table T1 which corresponds to the received index value.  
15

As is obvious above, two new fields are needed in the messages from A to B (i.e. in the input information) as compared to the single process system: the field indicating the index and the flag indicating whether process B can use direct pointing (i.e. read the pointer corresponding to the received index value and use the set of results addressed by that pointer).  
20

If there are more than two CAC processes within the network element, each process must keep a separate table for each such process from which it receives connection requests. For example, in the embodiment of Figure 5, process C has a separate table for process B. The last CAC process in the chain does not necessarily have to store the received index with the results, since it does not have to supply the index to any other CAC process.  
25

Instead of a pointer pointing the results, the table(s) T1 could also include the calculated results. However, the length of a pointer is only one word, whereas the results require much more memory space. In terms of memory space, it is therefore advantageous to store pointers in the table(s).  
30

By using "multistage" implementation as described above, only the first of the several CAC processes/units has to do the cache lookup. This makes the connection admission control of the network element faster and computationally less demanding.  
35

Although the invention has been described here in connection with the examples shown in the attached figures, it is clear that the invention is not limited to these examples, as it can be varied in several ways within the limits set by the attached patent claims. The following describes briefly some possible variations.

As mentioned above, the set of results calculated in the calculation block and stored in the cache can include one or more values indicating in some way the capacity requirement of the traffic source that sent the connection request. Thus, the number of values resulting from the calculation is not essential but rather that the parameters input to the calculation block and the value(s) resulting from the calculation are stored in the cache. Neither is it essential what these parameters are. Moreover, the parameters can be determined in many ways: all the parameters can be simply extracted from the connection request, for example, or one or more of the parameters can be calculated in the network element in response to the reception of the request.

When a parameter value in the linked list is compared to the value of the corresponding parameter in the request, an exact match is normally required in order to proceed further in that list. However, it is also possible that there is always a match when the difference between the values is smaller than a predetermined limit value. This interpretation of a "match" can also depend on the type of parameter.

In "multistage" implementation, it is also possible to combine the functions of the successive CAC processes so that some functions of the CAC unit are realized by a functional block which is common for several CAC processes. As discussed above, each unit has a cache of its own. However, some other parts of the CAC units can be implemented as centralized, especially because the CAC processes operate in succession. Thus, the functions of the main program block and the calculation block can be implemented by a single main program block or a single calculation block which is common for several or all CAC processes. In these cases a centralized main program block estimates the total capacity requirement and makes the decision for several or all CAC processes, and a centralized calculation block calculates the results for several or all CAC processes.

**Claims:**

1. A method of carrying out connection admission control in a communications network, the method comprising the steps of
  - receiving connection requests from traffic sources,
  - 5       - in response to a connection request received from a traffic source, determining a request-specific set of input data,
  - calculating a set of results by means of said set of input data, said set of results indicating a capacity requirement of said traffic source,
  - estimating a total capacity requirement on the basis of said set of
  - 10   results and the capacity requirement of existing connections, and
  - deciding on the acceptability of said connection request on the basis of the estimated total capacity requirement,
  - c h a r a c t e r i z e d by the further steps of
  - in response to said calculation, storing said set of input data and
  - 15   the corresponding set of results into a cache,
  - in response to said determination, searching in the cache for a set of data similar to the request-specific set of input data, and
  - using the stored set of results for estimating the total capacity when a similar set of data is found from the cache, whereby the calculation of
  - 20   the set of results is omitted when a similar set of data is found.
2. A method according to claim 1, wherein the connection requests belong to several service classes, c h a r a c t e r i z e d in that the sets of input data are stored class-specifically by forming a cache array for each service class, each cache array including a certain number of elements.
- 25   3. A method according to claim 2, c h a r a c t e r i z e d in that a pointer is stored in an individual element of a cache array, said pointer pointing to a data unit in a linked list of data units, in which list the set of results are stored in the last data unit and the set of input data is stored in the other data units.
- 30   4. A method according to claim 3, wherein the request-specific input data includes several traffic parameters of different types,
- c h a r a c t e r i z e d in that
- the values of the traffic parameters are stored in said other data units, one traffic parameter per data unit, and
- 35   a similar set of data is searched for by starting from a certain element of a class-specific cache array and proceeding in the list data unit by



data unit until either a traffic parameter value stored in the data unit differs from the value of the corresponding traffic parameter in the set of input data or the set of results is reached.

5 5. A method according to claim 3, wherein the set of input data includes several traffic parameters of different types,

characterized in that

the values of the traffic parameters are stored in said other data units, the value of one traffic parameter in each data unit, and

10 a similar set of data is searched for by examining first the first data unit in each class-specific list, then the second data units in those lists in which the value of the first parameter matches the value of the corresponding parameter in the request-specific set of input data, then the third data units in those lists in which the values of the first and the second parameter match the values of the corresponding parameters in the request-specific set of input  
15 data, etc.

6. A method according to claim 1, wherein the connection requests belong to different service classes and the request-specific set of input data includes several traffic parameters of different types, characterized in  
20 that a separate parameter array is formed for each of the service classes, an individual class-specific parameter array including  $N_i$  elements, where  $N_i$  is the number of possible values of a selected traffic parameter in service class  $i$  ( $i=1\dots N$ ), each element pointing to a cache array including a certain number of elements.

7. A method according to claim 6, characterized in that a  
25 pointer is stored in an individual element of a cache array, said pointer pointing to a data unit in a linked list of data units, in which list the set of results are stored in the last data unit and the values of all the other traffic parameters except the value of said selected parameter are stored in the other data units, with one parameter value per each data unit.

30 8. A method according to claim 1, characterized in that the sets of input data and the corresponding sets of results stored in the cache are updated on a FIFO basis by replacing the oldest pair of sets when a new set of input data and the corresponding set of results is stored in the cache.

9. A method according to claim 1, characterized by  
35 - monitoring how often each set of results is found from the cache,  
and

- in response to said monitoring, selecting the set of input data and the corresponding set of results to be replaced by a new set of input data and a new set of results.

10. A method according to claim 1, c h a r a c t e r i z e d by

5       - using several CAC processes within a network element which operate in a chain so that when a process has accepted the connection request for its part, the next process in the chain starts, whereby each process (1) stores said set of input data and the corresponding set of results into a cache provided for that process, (2) uses the stored set of results for estimating the total capacity when a similar set of data is found from the cache, (3)

10       estimates a total capacity requirement on the basis of said set of results and the capacity requirement of existing connections, and (4) decides for its part on the acceptability of said connection request on the basis of the estimated total capacity requirement, and

15       - accepting the connection request when all processes accept it.

11. A method according to claim 10, c h a r a c t e r i z e d in that in two successive CAC processes in said chain

- an identifier is stored for each set of results at the preceding process (A),

20       - the subsequent process (B) is informed of the identifier relating to the set of results obtained in the preceding process, and

- a flag is used for indicating to the subsequent process whether the preceding process found a similar set of data from its cache.

12. A method according to claim 11, c h a r a c t e r i z e d by

25       - calculating the set of results in the subsequent process when said flag indicates that the preceding process failed to find a similar set of data in its cache,

- storing at least said set of input data and the calculated set of results in the cache, and

30       - storing a pointer in a memory location defined by said identifier, said pointer pointing to the set of results stored in the cache of the subsequent process.

13. A method according to claim 12, c h a r a c t e r i z e d by further storing the identifier in the cache of the subsequent process.

35       14. A method according to claim 12, c h a r a c t e r i z e d by

- reading the pointer identified by said identifier when the preceding process indicates that it has found a similar set of data from its cache and
- using the set of results addressed by said pointer.

5 15. A connection admission control arrangement for a network element adapted to receive connection requests from traffic sources in a communications network, the arrangement comprising

- calculation means for calculating a set of results by supplying a set of input data to said calculation means, said set of results indicating the capacity needed by a traffic source,
  - 10 - means for determining said set of input data in response to a connection request,
  - estimation means for estimating a total capacity requirement on the basis of said set of results and the capacity requirement of connections existing in the network, and
  - 15 - decision means for deciding on the acceptability of said connection request on the basis of the total capacity requirement,
- c h a r a c t e r i z e d in that the arrangement further includes
- storing means for storing said set of input data and the corresponding set of results into a cache, said storing means being responsive to
  - 20 said calculation means, and
  - searching means for looking for a set of data similar to said set of input data from the storing means,
- said searching means being adapted to supply to the estimation means the set of results found in the storing means when a similar set of input data is found and to control the calculation means for calculating a set of results when no similar set of input data is found in the storing means.

25 16. A connection admission control arrangement according to claim 15, c h a r a c t e r i z e d in that the storing means includes

a cache array (CA1...CAN) for each service class used in the network, each cache array including a certain number of elements, whereby each

30 element can be empty or include a pointer pointing to a data unit in a linked list of data units, each linked list being empty or including a set of input data and the corresponding set of results.

35 17. A connection admission control arrangement according to claim 15, wherein the set of input data comprises a set of traffic parameters, c h a r a c t e r i z e d in that the storing means includes

a separate parameter array for each service class used in the network, an individual parameter array including  $N_i$  elements, where  $N_i$  is the number of possible values of a selected traffic parameter in service class  $i$  ( $i=1\dots N$ ), each element including a pointer pointing to a cache array including a certain number of elements, whereby each cache array element can be empty or include a pointer to a data unit in a linked list of data units, each linked list being empty or including all the other traffic parameters in the set of input data, except said selected traffic parameter, and the set of results corresponding to said traffic parameters.

10            18. A connection admission control arrangement according to claim 15, characterized in that

15            - the arrangement comprises several separate connection admission control units (1A, 1B, 1C), each unit including at least the storing means, whereby the total capacity requirement is estimated for each unit and a unit-specific decision is made on the acceptability of said connection request, and

20            - the connection admission control units are connected operably in series so that the connection request is forwarded to the next unit when the connection request has been accepted for the preceding unit, whereby the connection request is accepted in the network element when it has been accepted for all units.

19. A connection admission control arrangement according to claim 18, characterized in that each connection admission control unit further includes the calculation means, the estimation means and the decision means.

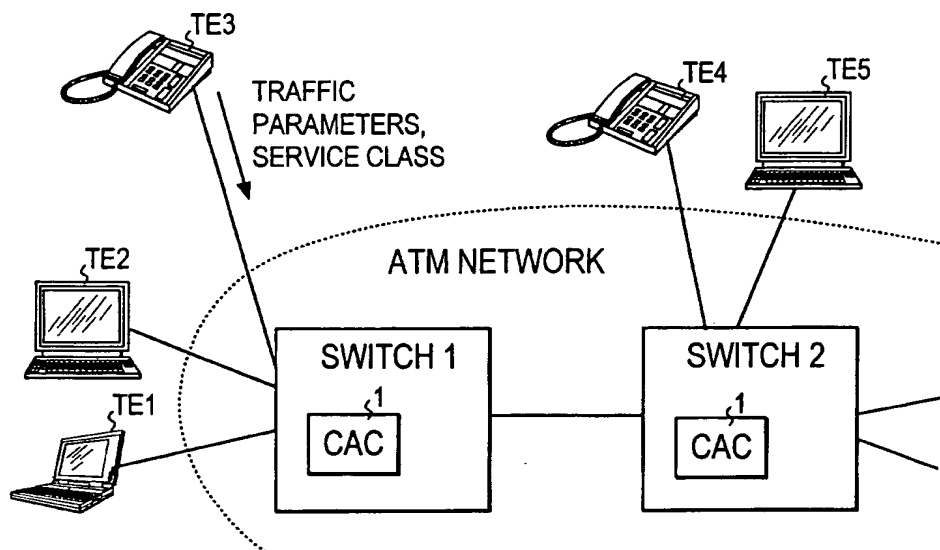


FIG. 1

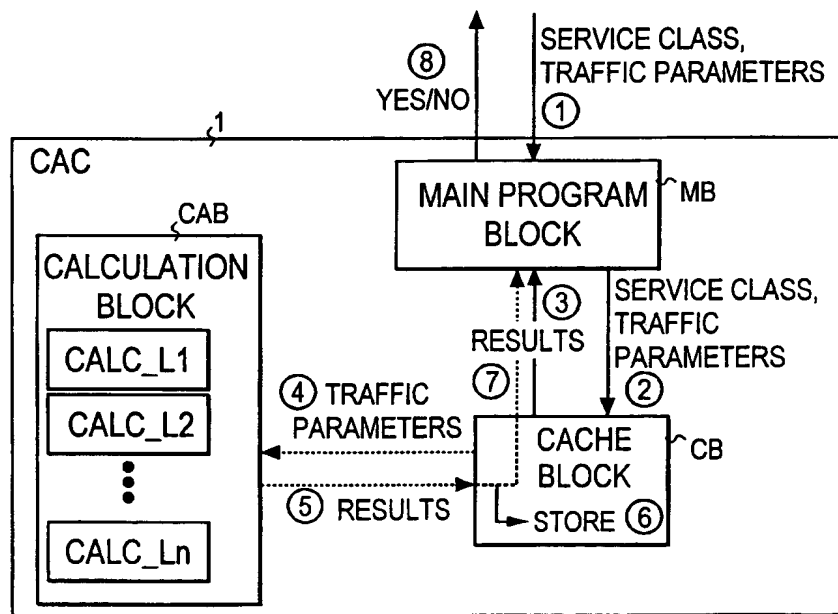


FIG. 2

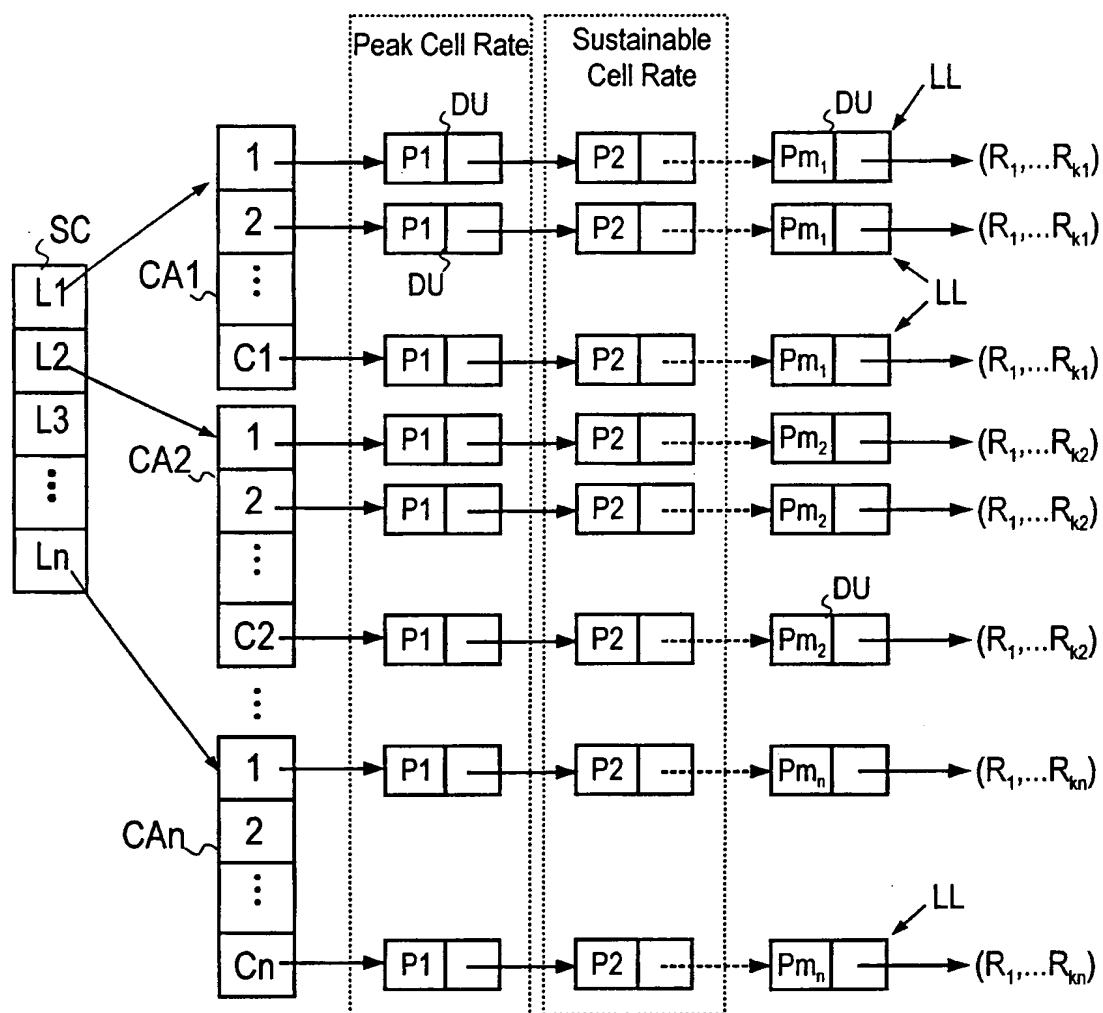


FIG. 3

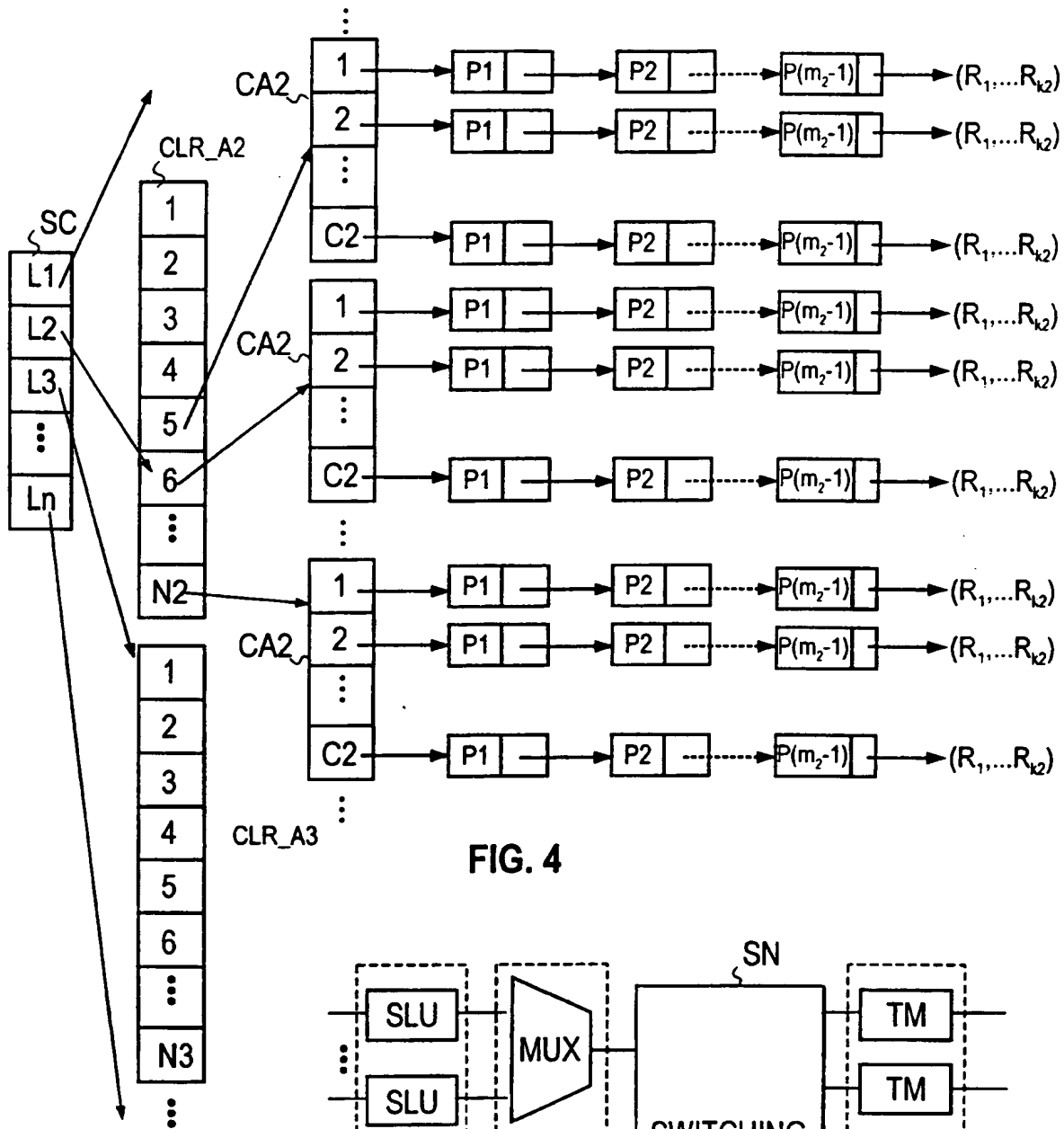


FIG. 5

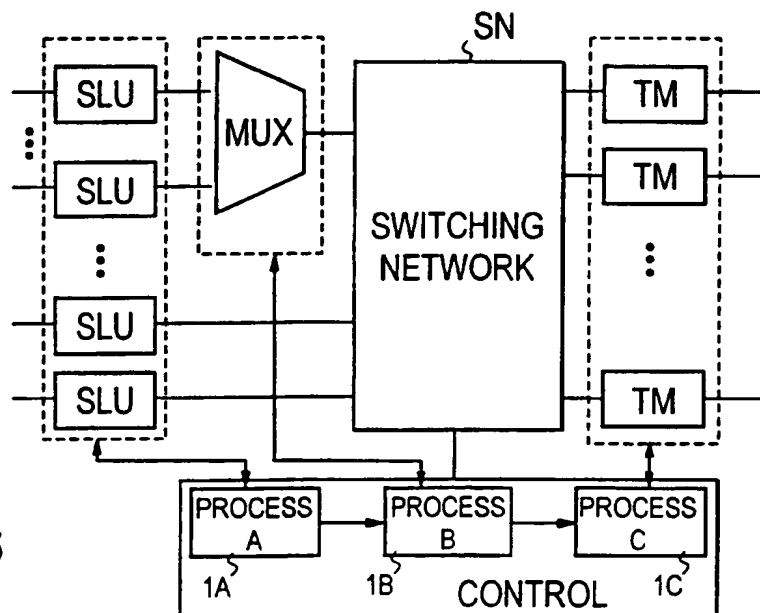


FIG. 6

